ESO: EVOLUTIONARY SPECTROGRAM OPTIMISATION FOR PASSIVE ACOUSTIC MONITORING

Ufuk Çakır
Interdisciplinary Center for Scientific Computing,
University of Heidelberg, Germany
ufuk.cakir@stud.uni-heidelberg.de

Lorène Jeantet
African Institute for Mathematical Sciences,
South Africa
lorene@aims.ac.za

Joel Lontsi
African Institute for Mathematical Sciences,
South Africa
joellontsi@aims.ac.za

Emmanuel Dufourq
African Institute for Mathematical Sciences,
South Africa,
African Institute for Mathematical Sciences,
Research and Innovation Centre, Rwanda
dufourq@aims.ac.za

ABSTRACT

Passive Acoustic Monitoring provides an important tool for wildlife monitoring. Deep Learning and the use of convolutional neural networks have become the common approach to create bioacoustic classifiers. However, in order to perform these tasks in real-time, standard approaches suffer from high model complexity and computationally expensive pre-processing steps (such as downsampling). In this paper we introduce a genetic algorithm named ESO, designed to optimise the input spectrogram size by focusing on specific regions. This approach allows for significant reduction in model complexity (91%) and inference time (70%) with minimal impact on the F1-Score (4%). We furthermore develop a simple-to-use Graphical User Interface and Python package to run the algorithm. All our code is available on GitHub[1].

1 INTRODUCTION

Wildlife monitoring is critical, especially for species which are declining at an alarming rate, or which are already on the brink of extinction (Almond et al., 2022). Passive acoustic monitoring (PAM) represents one method of monitoring wildlife by deploying multiple digital sound recorders, generally in remote areas, for extended periods. This method enables the study of species that are challenging to observe by identifying their vocalisations, or calls, within the acoustic recordings (Penar et al., 2020; Teixeira et al., 2019). Classification and detection algorithms are usually applied to the acoustic data after it has been retrieved and downloaded. This is primarily due to the fact that modern classification and detection algorithms use deep learning methodologies which are not easily implemented on recording devices. Developing real-time PAM systems remains an active area of research which offers great potential for immediate conservation actions to take place.

Deep learning algorithms, such as convolutional neural networks (CNNs), have become the common approach to solving classification tasks in bioacoustics (Stowell, 2022). This approach relies on pre-processing acoustic signals into spectrograms, which are visual representations of sounds used as inputs by the CNN. A

[1] https://github.com/ufuk-cakir/ESO

spectrogram displays the spectrum of frequencies of the acoustic signal over time. Spectrograms can pose a challenge when training CNNs in data scarcity situations. Large spectrograms result in a larger number of CNN parameters, which in turns hinders the implementation of CNNs on low-resource devices.

Typically, prior to the creation of a spectrogram, pre-processing techniques such as low-pass filtering and downsampling are applied to the audio signals. These operations are computationally expensive, and this issue is accentuated when using low-resource devices. As a result, the development of real-time PAM systems on low-resource devices requires the least amount of computation to be performed so that acoustic signals can be pre-processed and processed by the CNN rapidly in real-time.

As a means of contributing methodologies to facilitate real-time PAM systems, we propose an Evolutionary Spectrogram Optimisation (ESO) algorithm to reduce the size of spectrograms, and to reduce the total amount of computation performed in a PAM CNN classifier. ESO is based on a genetic algorithm (GA). A GA is an optimisation method inspired by natural selection, where a population of candidate solutions evolves over generations through nature-inspired processes like reproduction, selection, crossover, and mutation (Kumar et al., 2010; Eiben et al., 1994). In this study, we adapt the GA to reduce spectrogam representations to select and extract narrow bands within the spectrogram. The bands serve as input to a CNN model, i.e. the bands represent extracted portions of the original spectrogram. ESO thus optimises the selection and size of each of these bands to maximise classification score and minimise CNN parameters. Additionally, in ESO, spectrograms are generated without low-pass filtering and downsampling. When compared to using the full spectrogram, our findings reveal that ESO can reduce the computation time by 70%, reduce the CNN parameters by up to 90%, and significantly improve the inference time. ESO provides the ability to implement real-time PAM systems which could be implemented directly in field studies on low-resource edge devices.

2 METHODS

2.1 HAINAN GIBBON DATASET AND BASELINE MODEL

To assess the efficiency of ESO to automatically identify bands of interest within spectrograms for bioacoustic classification, we used the publicly available Hainan gibbons dataset (Dufourq et al., 2021). This dataset consists of 220 hours of annotated acoustic files recorded at 9600 Hz using passive acoustic recorders. Using this dataset, ESO was tasked to optimise the bands which would serve as input to the CNN classifier. The CNN output represents the presence or absence of Hainan gibbon calls from large audio files, of up to 8 hours in duration.

Dataset Creation – To compare ESO with the traditional approach of involving pre-processing steps (low-pass filtering and downsampling), we constructed two datasets. One dataset consists of spectrograms generated after downsampling and low-pass filtering similar to Dufourq et al. (2021). The other dataset consists of spectrograms generated directly from the raw acoustic files. In the first case, the acoustic files were downsampled to 4800 Hz, and high-frequency noise was eliminated by applying a low-pass filter to each recording with a cutoff rate of 2000 Hz. Subsequently, the files were segmented into four-second windows according to manual annotation and transformed into spectrograms, resulting in images sized at 128×76 pixels. In the second case, the files were only segmented into windows before being transformed into spectrograms sized at 128×151.

We randomly split both datasets into train-validation-test (60 - 20 - 20). We obtained $17,130$ four-second windows ($7,978$ gibbon calls, $9,152$ no-gibbon calls) for each training dataset, $5,319$ windows for the validation dataset ($2,590$ gibbon calls, $2,729$ no-gibbon calls) and $5,774$ windows for the testing dataset ($3,423$ gibbon calls, $2,351$ no-gibbon calls). We performed data-augmentation similarly to Dufourq et al. (2021) to obtain a balanced training dataset ($9,152$ gibbon calls, $9,152$ no-gibbon calls).

Baseline Model – For the automated classification of Hainan gibbon calls, and as the baseline model for this study, we used a similar architecture as in Dufourq et al. (2021). The baseline model consists of a simple CNN architecture, featuring one convolutional layer with 8 filters of size 8×8 and ReLU activations, followed by max pooling (4×4), a flattening operation, and two fully-connected layers (32 ReLU units and 2 softmax units, respectively). The final softmax layer provides the probability that the input spectrogram contains a gibbon call or not. For the baseline CNN we used the dataset created with downsampling and low-pass filtering. We trained the baseline CNN on the training data and evaluated its performance on the validation data by calculating the F1-score. This F1-score, together with the number of trainable parameters of the baseline model, was recorded and used as reference values by the ESO algorithm as a means of comparison.

2.2 ESO: Evolutionary Spectrogram Optimisation

Genetic Algorithm – The GA begins by creating an initial population of randomly initialised solutions for the problem to be solved. Each solution, also known as an individual, is represented by a chromosome. A chromosome is defined by a specified number of genes, where each gene encodes the information for one specific parameter of the solution. A new generation is produced from the initial population, with each individual being evaluated through a fitness function. Similar to natural selection, only the individuals with the best fitness are selected to form the new population.

Genetic Operators – Chromosomes for the next generation, also called offspring, are created using a reproductive process that includes natural genetic operations, such as crossover and mutation, to generate potential new solutions and explore other combinations of genes. Mutation creates a new chromosome by adding a small random variation to one of the genes inside a selected chromosome. Reproduction copies a chromosome from the current generation to the next one, whereas crossover creates two new chromosomes by exchanging genes from the parents. The process of evolution is then iterated through a predetermined number of generations or until some end condition is satisfied.

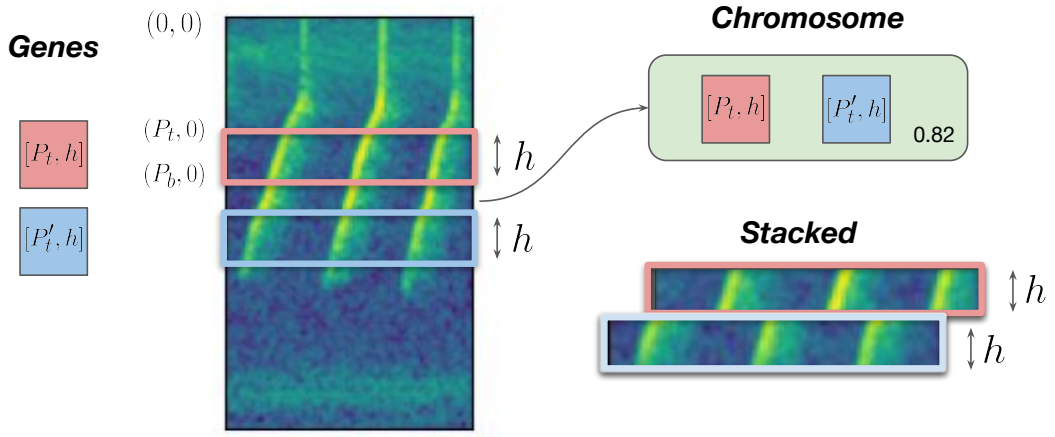

Figure 1: The image shows a four-second window spectrogram with a Hainan gibbon vocalisation. Each *gene* encodes one band defined by the band position P_t, P'_t (origin top left) and band height h. A *chromosome* as a collection of genes defines which bands to extract from the spectrogram images. The extracted bands are then stacked, as shown in the bottom right corner. The effectiveness of each chromosome to result in the best classification score is evaluated by the fitness function (bottom right of the chromosome), as defined in Equation 1.

ESO Chromosome – Each gene within ESO encodes the information of one band in the spectrogram image, which is uniquely defined by its band position and band height (measured in pixels). We define a band as a horizontal region constructed from two points, P_t being the top of the band, and the $P_b = P_t + h$ being the bottom and h the height of the band. Positions P_t and P_b are selected between the origin to the bottom left of the spectrogram. Figure 1 shows two bands within a chromosome. An ESO chromosome thus outputs various extracted bands from a spectrogram, which are stacked on top of each other. Thus, when applying one chromosome to a training dataset, the chromosome produces a modified (compressed) dataset which can then be used to train a CNN. For each chromosome, we trained a CNN on the stacked extracted bands of the spectrograms (bottom right of Figure 1), using the training dataset generated without low-pass filtering and downsampling. We used the same architecture as the baseline model, but with a smaller input shape (given the new stacked dataset), and hence less trainable parameters. We calculated the F1-score on the validation dataset and evaluated each chromosome using the fitness function defined by:

$$\text{Fitness} = -\lambda_1 \frac{\text{F1}_{\text{base}} - \text{F1}_{\text{chromosome}}}{\text{F1}_{\text{base}}} + \lambda_2 \frac{p_{\text{base}} - p_{\text{chromosome}}}{p_{\text{base}}} \tag{1}$$

where λ_1, λ_2 are hyper-parameters, *F1* represents the F1-score on the validation set obtained by the baseline model and by the chromosome and p is the number of trainable parameters of the chromosome and baseline model.

The proposed fitness function enables ESO to optimise a trade-off between model performance (F1-score) and model complexity (number of trainable parameters) while always comparing to the baseline model. The hyper-parameters λ_1 and λ_2 influence this trade-off.

ESO Algorithm – Each run of the ESO algorithm consists of randomly initialising the first population, training all chromosome CNNs on the training dataset and evaluating their performance on the validation dataset to calculate their fitness. The next generation of chromosomes is then created using the selection and genetic operators to explore the space of possible band combinations. In this context, mutation introduces a random shift from a predefined range to the band positions, while crossover exchanges randomly selected bands from two parent chromosomes. The best chromosome in terms of fitness value is saved at each generation, and the best chromosome over all generations is considered as the final solution.

We trained both the baseline model, and the best ESO chromosome for 40 epochs using the Adam optimiser (Kingma & Ba, 2017) and cross entropy loss, and we evaluated both approaches on the test set to get our final results. To contribute towards accessible tools, we enabled ESO to be executed using a Python package, or through a Graphical User Interface with full control over hyper-parameters and a built-in Tensorboard to check the training progress.

3 RESULTS

We ran the ESO algorithm on the Hainan gibbon dataset and averaged the results across ten runs, by keeping all hyper-parameters (see Appendix A.1) fixed (e.g 10 generations, 20 chromosomes per population, two to six genes per chromosome). The best chromosome across all generations was obtained and applied to the test set. Similarly, the baseline was applied to the test set, and we compared the two approaches.

The findings from Table 1 reveal that on average, ESO decreases the number of trainable parameters by 90%, the input image size by 50% and inference time by 70% with a trade-off by 4% lower F1-Score. This result shows that ESO can significantly boost inference time and reduce model complexity, both of which are the limiting factors of common passive acoustic monitoring approaches in the context of real-time wildlife monitoring. When using 10 generations and a population of 20 chromosomes, ESO took approximately 1 h when executed on consumer GPU hardware (NVIDIA GeForce GTX 1050 Ti, 4 GB) which shows that ESO

Table 1: Average performance metrics of models over ten runs with a population size of 20. Note that inference is the time it took to process the raw audio (40 hours) and get predictions. For the baseline model this includes downsampling and low-pass filtering the raw audio files. While ESO had a 4.23% drop in F1-Score, there were significant reductions in trainabale parameters, total computation performed, and inference time – thus revealing that ESO contributes towards real-time PAM systems.

Metric	**Baseline**	**ESO**	**Difference (%)**
Accuracy	0.96	$0.94_{\pm 0.00}$	$-2.01_{\pm 0.00}$
Sensitivity	0.95	$0.92_{\pm 0.01}$	$-3.16_{\pm 1.05}$
Specificity	0.97	$0.95_{\pm 0.01}$	$-2.06_{\pm 1.03}$
Precision	0.92	$0.88_{\pm 0.02}$	$-4.35_{\pm 2.17}$
F1-Score	0.94	$0.90_{\pm 0.01}$	$\mathbf{-4.26}_{\pm 1.06}$
Parameters	$132,234$	$11,863_{\pm 247}$	$\mathbf{-91.03}_{\pm 0.19}$
Image Size	$9,728$	$4,817_{\pm 800}$	$\mathbf{-50.48}_{\pm 8.22}$
Inference time [s]	376.1	106.9	$\mathbf{-71.58}$

does not require high-end GPU resources to be effective, hence making it a powerful and accessible tool for researchers.

4 Conclusion

Our study has shown that an evolutionary optimisation approach coupled with deep learning can significantly reduce the model complexity and the inference time while achieving comparable performance with a model trained on spectrograms generated from pre-processed audio files. These results therefore suggest that the ESO method could pave the way to the development of real-time monitoring techniques which could be performed in the field on low-powered edge devices. The development of a GUI aims to provide a powerful and simple to use tool for the ecology community. Future experiments will take into account datasets with animal vocalisations in different frequency regions and further improvements of the algorithm such as parallelisation if multiple GPUs are available for training.

5 Acknowledgments

UC was supported by the HeiAIMS partnership with support from Baden-Württemberg-STIPENDIUM and Baden-Württemberg Stiftung. ED was funded by a grant from the Carnegie Corporation of New York. LJ was funded by a grant from the International Development Research Centre, Ottawa, Canada, www.idrc.ca, and with financial support from the Government of Canada, provided through Global Affairs Canada (GAC), www.international.gc.ca. The statements made and views expressed are solely the responsibility of the author(s). JL was funded by a grant from the Department of Science and Innovation.

References

REA Almond, M Grooten, D Juffe Bignoli, and T Petersen. Living planet report 2022–building a nature-positive society. *World Wildlife Fund*, 2022.

Emmanuel Dufourq, Ian Durbach, James P. Hansford, Amanda Hoepfner, Heidi Ma, Jessica V. Bryant, Christina S. Stender, Wenyong Li, Zhiwei Liu, Qing Chen, Zhaoli Zhou, and Samuel T. Turvey. Auto-

mated detection of Hainan gibbon calls for passive acoustic monitoring. *Remote Sensing in Ecology and Conservation*, 7(3):475–487, 2021. ISSN 20563485. doi: 10.1002/rse2.201.

Agoston E Eiben, P-E Raue, and Zs Ruttkay. Genetic algorithms with multi-parent recombination. In *International conference on parallel problem solving from nature*, pp. 78–87. Springer, 1994.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Manoj Kumar, Mohamed Husain, Naveen Upreti, and Deepti Gupta. Genetic Algorithm: Review and Application. *International Journal of Information Technology and Knowledge Management*, 2(2):451–454, 2010. doi: 10.2139/ssrn.3529843.

Weronika Penar, Angelika Magiera, and Czesław Klocek. Applications of bioacoustics in animal ecology. *Ecological Complexity*, 43:100847, 2020. ISSN 1476-945X. doi: https://doi.org/10.1016/j.ecocom.2020.100847. URL https://www.sciencedirect.com/science/article/pii/S1476945X19301606.

Dan Stowell. Computational bioacoustics with deep learning: a review and roadmap. *PeerJ*, 10:e13152, March 2022. ISSN 2167-8359. doi: 10.7717/peerj.13152. URL http://dx.doi.org/10.7717/peerj.13152.

Daniella Teixeira, Martine Maron, and Berndt J. van Rensburg. Bioacoustic monitoring of animal vocal behavior for conservation. *Conservation Science and Practice*, 1(8):e72, 2019. doi: https://doi.org/10.1111/csp2.72. URL https://conbio.onlinelibrary.wiley.com/doi/abs/10.1111/csp2.72.

A APPENDIX

A.1 HYPER-PARAMETERS

We choose following hyper-parameters for our experiments:

```
settings = {
    "algorithm": {"max_generations": 10},
    "genetic_operator": {
        "mutation_rate": 0.1,
        "crossover_rate": 0.8,
        "reproduction_rate": 0.1,
        "mutation_height_range": 0,
        "mutation_position_range": 10,
    },
    "selection_operator": {"tournament_size": 10},
    "data": {
        "force_recreate_dataset": False,
        "keep_in_memory": False,
        "train_size": 0.8,
        "test_size": 0.2,
        "reshuffle": False,
        "positive_class": "gibbon",
        "negative_class": "no-gibbon",
    },
    "preprocessing": {
        "lowpass_cutoff": 2000,
        "downsample_rate": 4800,
        "nyquist_rate": 2400,
        "segment_duration": 4,
        "nb_negative_class": 20,
        "file_type": "svl",
        "audio_extension": ".wav",
        "n_fft": 1024,
        "hop_length": 256,
        "n_mels": 128,
        "f_min": 4000,
        "f_max": 9000,
    },
    "population": {"pop_size": 20},
    "gene": {
        "min_position": 0,
        "max_position": -1,
        "band_height": 11,
    },
    "chromosome": {
        "min_num_genes": 1,
        "max_num_genes": 4,
        "lambda_1": 0.7,
        "lambda_2": 0.3,
    },
    "model": {
        "optimizer_name": "adam",
        "loss_function_name": "cross_entropy",
        "num_epochs": 40,
        "batch_size": 64,
        "learning_rate": 0.001,
        "shuffle": True,
        "metric": "f1",
    },
}
```

A.2 GRAPHICAL USER INTERFACE

An overview of how the Graphical User Interface for the ESO algorithm is shown in Figure 2.

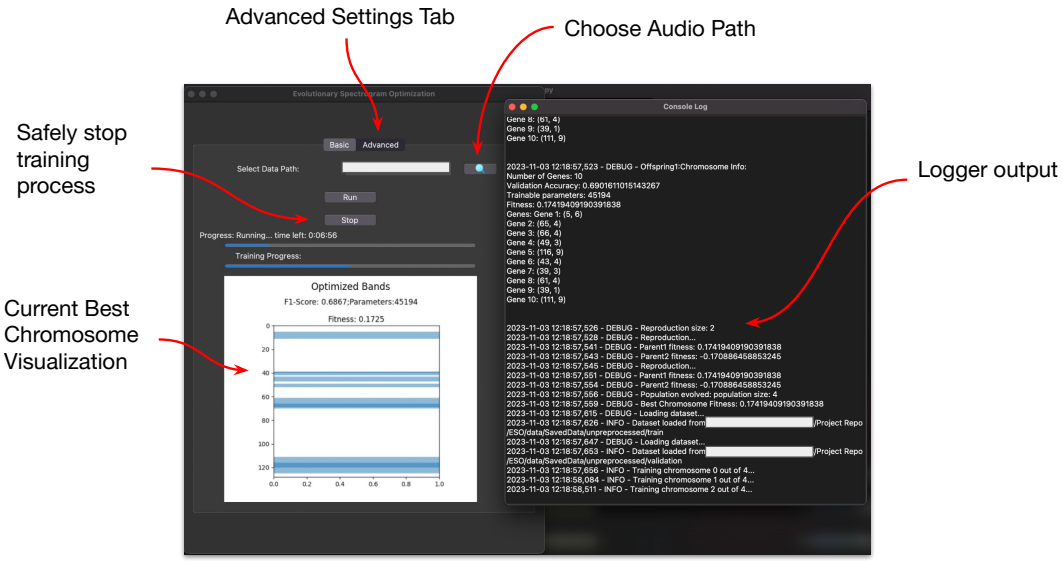

Figure 2: Graphical User Interface for the ESO Algorithm

The GUI additionally has the option to launch a TensorBoard session to track training progress across different runs and tune hyperparameters in the parallel coordinates view.